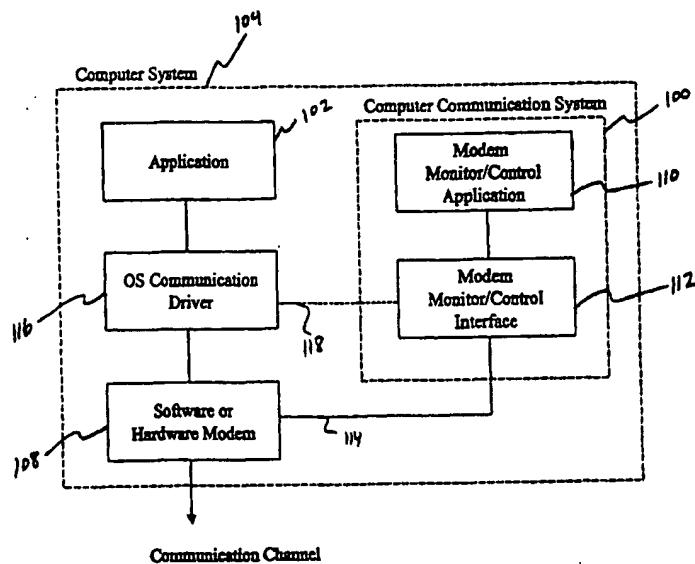




## INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 :  H04M 11/06, H04L 12/26, 12/24		A1	(11) International Publication Number: <b>WO 99/45694</b>  (43) International Publication Date: 10 September 1999 (10.09.99)
(21) International Application Number: PCT/US99/04690			(81) Designated States: AL, AM, AU, AZ, BA, BB, BG, BR, BY, CA, CN, CU, CZ, EE, GE, GH, GM, HR, HU, ID, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, RO, RU, SD, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, UZ, VN, YU, ZW, European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).
(22) International Filing Date: 4 March 1999 (04.03.99)			
(30) Priority Data:			Published
60/076,784	4 March 1998 (04.03.98)	US	With international search report.
09/154,643	17 September 1998 (17.09.98)	US	Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.
09/192,627	16 November 1998 (16.11.98)	US	
(71) Applicant: CONEXANT SYSTEMS, INC. [US/US]; 4311 Jamboree Road, Newport Beach, CA 92660-3095 (US).			
(72) Inventors: COLLIN, Zeev; Shapira Street 9/2, 46406 Herzliya (IL). TAMIR, Tal; Bloch Street 21A, 53229 Givatayim (IL).			
(74) Agent: SCOTT, Russell, C.; Akin, Gump, Strauss, Hauer & Feld, LLP, Suite 1900, 816 Congress Avenue, Austin, TX 78701 (US).			

(54) Title: METHOD AND APPARATUS FOR MONITORING, CONTROLLING, AND CONFIGURING LOCAL COMMUNICATION DEVICES



## (57) Abstract

A communication system for monitoring and/or controlling communication parameters of a communication device. The communication system monitors a communication channel that is created when the communication device connects to a network, controls the communication device as it operates on the network, and configures the communication device. The communication device is commonly a modem and is communicatively coupled to the network to carry out ongoing communications between the modem and the network through the communication channel. Further, a software module is associated with the modem, and the software module accesses the internal settings of the modem via the communication channel (if necessary) and performs operations such as monitoring, controlling, and configuring the modem (or other communication device) using the internal settings of the modem.

**FOR THE PURPOSES OF INFORMATION ONLY**

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece	ML	Mali	TR	Turkey
BG	Bulgaria	HU	Hungary	MN	Mongolia	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MR	Mauritania	UA	Ukraine
BR	Brazil	IL	Israel	MW	Malawi	UG	Uganda
BY	Belarus	IS	Iceland	MX	Mexico	US	United States of America
CA	Canada	IT	Italy	NE	Niger	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NL	Netherlands	VN	Viet Nam
CG	Congo	KE	Kenya	NO	Norway	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NZ	New Zealand	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	PL	Poland		
CM	Cameroon	KR	Republic of Korea	PT	Portugal		
CN	China	KZ	Kazakhstan	RO	Romania		
CU	Cuba	LC	Saint Lucia	RU	Russian Federation		
CZ	Czech Republic	LI	Liechtenstein	SD	Sudan		
DE	Germany	LK	Sri Lanka	SE	Sweden		
DK	Denmark	LR	Liberia	SG	Singapore		

**INTERNATIONAL APPLICATION**  
**UNDER THE PATENT COOPERATION TREATY**

5

**TITLE:** **Method and Apparatus for Monitoring, Controlling, and Configuring Local Communication Devices**

10 A portion of the disclosure of the patent document contains material which is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by any-one of the patent document or the patent disclosure, as it appears in the publication of the International Application, but otherwise reserves all copyright rights whatsoever.

**1. Technical Field**

15 The present invention relates to communication systems and more particularly to a computer communication system that, among other things, monitors, controls, and configures communication parameters of the computer communication system while one computer system communicates with another computer system.

20 **2. Background Art**

In traditional implementations, control and monitoring of computer communication systems primarily concern monitoring and controlling internal parameters of modems and are performed through the use of modem control strings such as "AT commands". AT commands require a user to switch the modem from data to command mode so that the modem can be controlled with AT commands. Thus, AT commands interfere with the typical data flow of the modem and the commands do not reflect the true state of the modem in real time. Of note, in some traditional hardware modem implementations, limited control and status monitoring capabilities are obtained through adding special non-standard hardware interfaces. However, these special hardware interfaces are a relatively expensive solution to the problem of real time 25 modem monitoring and the usage is limited due to its complexity.

If the user chooses not to add the additional network equipment to retrieve the modem information, the user is forced to rely on verbal guidance from another person, such as a support technician, located at a second modem site. This support technician views the parameters of the modem connection from their end of the connection, performs a modem diagnosis based on 5 available resources, and reports configuration options to the user for manual modem control and monitoring. Clearly, this process for modem monitoring and control is unsatisfactory because, among other things, the process requires detailed and easily misunderstood verbal instructions for modem configuration, the process requires the modem to be switched from data to command mode to enter the diagnostic commands for modem configuration, and at least two people are 10 required to diagnose and configure a single modem. Thus, the monitor and configuration process is time consuming and frustrating for those involved.

Of current interest is a computer communication system that overcomes the disadvantages of the related art. Among other advantages and benefits, the computer communication system according to the principles of the present invention monitors, controls, 15 and configures communication parameters of the computer communication system while one computer system communicates with another computer system. In one embodiment, the computer communication system provides a modem monitor and control system that provides modem monitoring and control without requiring user interaction or switching the modem between data and command modes.

DISCLOSURE OF THE INVENTION

Various aspects of the present invention can be found in a communication system for monitoring and controlling communication parameters of a communication device. The communication system includes a communication device (often a modem) and a communication channel. The communication device, e.g., the modem, has internal settings representing communication parameters and is communicatively coupled to the communication channel to carry out ongoing communications from the modem through the communication channel. The communication system also includes a software module that is associated with the modem. The software module accesses the internal settings of the modem via the communication channel and performs diagnostics using the internal settings of the modem.

The software module of the communication system also includes a modem interface that interacts with the software module and assists the software module in performing diagnostics using the internal parameters of the modem. The software module accesses the communication channel transparently to the ongoing communications from the modem when the software module performs the diagnostics using the internal parameters of the modem. The software module may also access the communication channel without detrimentally affecting the ongoing communications across the communication channel. The software module may also perform diagnostics using the internal parameters of the modem via the same communication channel that is used to carry out ongoing communications to and from the modem.

In other embodiments, the diagnostics performed by the software module of the communication system comprises monitoring a data stream in the communication channel. The diagnostics performed by the software module may comprise configuring the internal settings of the modem based on information obtained regarding a data stream to and from the modem. The diagnostics may also comprise controlling the internal settings of the modem according to information obtained regarding a data stream to and from the modem. The communication system may also include a user interactive interface in the software module

for diagnostics and a plurality of software modules associated, respectively, with each of a plurality of modems.

The modem of the communication system is frequently communicatively coupled to the communication channel and thus to a network. The network is selected from the group 5 consisting of at least a local area network, a wide area network, and a global area network.

Various other aspects of the present invention can be found in a communication system comprising a first communication device having internal parameters, a second communication device having internal parameters and being communicatively coupled to the first communication device, a communications link that passes a data stream between the first 10 communication device and the second communication device, and a module associated with the communications link that adjusts the internal parameters of the first communication device based on characteristics of the internal parameters of the first communication device, the second communication device, or both.

The module of the communication system may also include a communication 15 interface that interacts with the communications link such that the module operates transparently to the data stream of the communications link. Further, the first communication device may comprise a local communication device and the second communication device comprises a remote communication device. The first communication device and the second communication device, respectively, often comprise a first modem and a second modem. 20 The communications link often operates on a network being selected from the group consisting of a local area network, a wide area network, and a global area network. In many embodiments, the communication system comprises a computer communication system and the module adjusts the internal parameters of the first communication device based on characteristics of the internal parameters of the first communication device, the second 25 communication device, or both.

Still further aspects of the present invention are found in a method for adjusting parameters of a communication system. The method comprises establishing a communications link between a first communication device and a second communication

device, each communication device having internal parameters influencing communication protocols on the communications link; obtaining a software module for interacting with the communications link; retrieving, with the software module, characteristics of the first communication device based on the internal parameters of the first communication device, 5 the second communication device, or both; and adjusting the internal parameters according to the retrieved characteristics to optimize communication between the first and the second communication devices on the communications link.

Adjusting the internal parameters may include adjusting the internal parameters of the first communication device, and, in addition, adjusting the internal parameters may include 10 monitoring or controlling the internal parameters of the first communication device. Further, retrieving characteristics of the first communication device may comprise retrieving the characteristics transparently to the data passing through the communications link and/or retrieving the characteristics such that the data passing through the communications link is not detrimentally affected.

15 In other aspects of the present invention, the communication system monitors a communication channel that is created between a first modem and a second modem and controls the first modem by adjusting internal settings of the first modem that represent communication parameters. The second modem is communicatively coupled to the first modem to carry out ongoing communications between the first modem and the second 20 modem through the communication channel. Further, a software module is associated with the first modem, and the software module accesses the internal settings of the first modem, via the communication channel or otherwise, and performs diagnostics using the internal settings of the first modem. Of course, the software module could access the internal settings of the first modem directly to perform diagnostics using the internal settings of the first 25 modem.

The software module of the communication system typically includes a modem interface that interacts with the software module and assists the software module in performing diagnostics using the internal parameters of the first modem. Also, whether monitoring or controlling the first modem, the software module accesses the communication

channel transparently to the ongoing communications between the first modem and the second modem when the software module performs the diagnostics. Further, the software module accesses the communication channel without detrimentally affecting the ongoing communications between the first modem and the second modem.

5 In another embodiment, the software module configures the first modem using the internal parameters of the first modem.

10 The diagnostics performed by the software module of the communication system include monitoring a data stream in the communication channel in view of the internal settings of the first modem. Further, the diagnostics performed by the software module comprise configuring the internal settings of the first modem based on information obtained 15 regarding the data stream between the first modem and the second modem. In addition, the diagnostics performed by using the software module comprise controlling the internal settings of the first modem according to information obtained regarding the data stream between the first modem and the second modem.

15 It should be noted that the software module may include either a user interactive interface for diagnostics, or an automatic interface for diagnostics that requires no further user interaction. Further, the communication system may include a plurality of software modules being associated, respectively, with each of a plurality of modems. Regardless of the number 20 of modems in the communication system, the modems are communicatively coupled via a network. The network is typically selected from the group consisting of a local area network, a wide area network, and a global area network, however, the network may include any combination of a local, wide, or global area network. In other words, the network could operate according to almost any existing network protocol, e.g., a peer-to-peer network, a transmission control protocol/Internet protocol network (TCP/IP), etc.

25 In another embodiment, the present invention can be described as a communication system comprising a first communication device having internal parameters; a second communication device having internal parameters and being communicatively coupled to the first communication device; a communications link that passes a data stream between the first

communication device and the second communication device; and a module associated with the communications link that adjusts the internal parameters of the first communication device based on characteristics of the internal parameters of the first communication device.

In this embodiment, the module may include a communication interface that interacts 5 with the communications link such that the module operates transparently to the data stream of the communications link. Further, the first communication device may be a local communication device and the second communication device may be a remote communication device. In addition, similar to the first embodiment, the communications link operates on a network such as a local area network, a wide area network, or a global area 10 network or a combination thereof. In many embodiments, the communication system is designed for modems operating in a computer communication system. Thus, to assist in understanding the principles according to the present invention, the exemplary embodiments are generally described using computer systems communicating with modems.

#### BRIEF DESCRIPTION OF DRAWINGS

15 A better understanding of the present invention can be obtained when the following detailed description of the preferred embodiment is considered in conjunction with the following drawings, in which:

20 Figure 1 is a block diagram of an exemplary computer communication system according to the principles of the present invention wherein the system is associated with an application for providing a computer system access to a communication channel via a modem.

Figure 2 is a block diagram of an exemplary modem monitor/control interface of the computer communication system of Figure 1.

25 Figure 3 is a block diagram illustrating an exemplary modem for operation with the computer communication system of Figure 1.

Figure 4 is a block diagram of an exemplary computer communication system for monitoring and controlling both a local modem and a remote modem over a telephone line.

Figure 5 is a block diagram of exemplary computer communication systems operating modem monitor/control applications, respectively, on both a client modem and a server 5 modem in a peer-to-peer network.

Figure 6 is a block diagram of exemplary computer communication systems operating modem monitor/control applications, respectively, on both a local computer system and a remote computer system, the systems communicating across the Internet.

Figure 7 is a block diagram of an exemplary computer communication system 10 operating according to simple network management protocol (SNMP) parameters such that a management application provides for trouble shooting of a local modem from remote locations.

MODE(S) FOR CARRYING OUT THE INVENTION

Figure 1 is a block diagram of an exemplary computer communication system 100 that operates according to the principles of the present invention. For ease of understanding, the system 100 is associated with a computer software application 102 for providing a computer system 104 access to a communication channel 106 via a communication device such as a modem 108. The computer software application 102 is commonly a typical computer telecommunications application such as a "web browser", viz., Netscape™, Internet Explorer™, etc., or a modem utility, viz., Procomm™, etc. In short, the computer software application 102 utilizes the modem 108 capabilities to communicate with other modems through the communication channel 106. While the computer software application 102 uses the modem 108 to communicate with other modems, the computer communication system 100 examines the modem parameters of the modem 108 to determine if the modem configuration needs to be modified to attain optimal performance through the communication channel 106. As stated, the computer communication system 100 is an exemplary embodiment that is used to facilitate understanding of the principles according to the present invention. It should be understood that the present invention applies equally well to communication systems that operate with communication devices other than modems. However, for ease of understanding, the present invention will be described relative to computer communication systems using modems as the communication devices.

The computer communication system 100 includes a modem monitor/control application 110 that performs diagnostics on the modem 108 through a modem monitor/control interface 112 (the modem monitor/control application 110 and the modem monitor/control interface 112 sometimes collectively referred to herein as a "software module"). Thus, diagnostics can be performed on the "local" modem 108. Advantageously, some of the diagnostics can also occur transparently to ongoing communications in the communication channel 106. Thus, the modem communication connection, a.k.a., the "data stream", of the modem 108 can pass through the communication channel 106 without being detrimentally affected during diagnostics. Further, the diagnostics can be performed via user interaction through the modem monitor/control application 110 or, alternatively, certain diagnostics can be automated and performed independently of user interaction through the

application 110. As stated, if changes in the modem parameters are required to obtain optimal performance in the modem 108, some of the changes can be made without interruption in the data stream. Of course, the modem 108 could be a software modem or a hardware modem or any combination thereof, a pure software modem being defined as a 5 modem implemented entirely in software and relying on a computer's processor to modulate and demodulate signals. Of note, although graphical line 114 represents direct coupling of the modem monitor/control interface 112 with the modem 108, the modem monitor/control interface 112 could instead be directly coupled to an operating system communication driver 116 as represented by dashed line 118. Further, the term "diagnostics", as used herein, refers 10 at least to monitoring, controlling, or configuring a modem.

Figure 2 is a block diagram of the exemplary modem monitor/control interface 112 of the computer communication system 100. The modem monitor/control interface 112 includes a modem monitor/control application programming interface (API) 200, a modem monitor/control data link library (DLL) 202 that operates similarly to standard DLL software components, and a modem monitor/control driver 204 that operates similarly to standard software drivers. The API 200 provides code for monitoring and controlling a software modem while the modem is running or passing a data stream (see Appendixes A, B, and C). API 200 provides an easy method to write applications that provide various diagnostics that monitor parameters that change in real time (such as MSE, baud rate, echo canceller 15 efficiencies, etc.) as well as enabling the writing of applications that allow internal parameters to be controlled while a telephony session is in progress. The API 200 can also provide easy means for field support by looking at various parameters and causing the modem to dump data into a file to be investigated later. Further, trouble shooting can be performed by changing various parameters while a data stream is running through the 20 modem. Of note, in a preferred embodiment, the API 200 operates asynchronously and in parallel with the ordinary modem operation and does not interfere with the data stream. Thus, API 200 provides a true view of the modem parameters and does not slow the data 25 transfer process.

Appendices A, B, and C include exemplary embodiments of code portions of the API 200 and include three functions that could be considered the backbone of the API 200. First, the ModemConfigure function configures parameters within the modem and should be called only before the modem is activated. Second, the ModemControl function changes parameters 5 within the modem to control the modem's operation and can be called during modem operation. Finally, the ModemMonitor function returns the current value of a parameter or set of parameters within the modem and can also be called during modem operation. The first parameter of the above functions is a code indicating which parameter (or parameter set) to monitor or change. The codes can be easily extended from time to time to provide 10. additional visibility and control options for the modem. The same interfaces apply for additional parts of the modem such as speakerphone, tone detection/generation, etc. Thus, the computer communication system 100 is extendable and easy to use and can be used to monitor and control a modem without interfering with the ordinary operation of the modem. Further, the computer communication system 100 provides an easy method to develop 15 applications for modem diagnostics and trouble shooting.

Figure 3 is a block diagram illustrating the exemplary modem 108 for operation with the computer communication system 100 that is associated with a computer system 104 for accessing a network. The exemplary modem 108 includes a port driver 300, a controller 302, a data pump abstraction layer (DPAL) 304, an advanced modem operation scheduler (AMOS) 20 306, a sample buffer management module 308, a hardware interface 310, and signal processing tasks 312. Of course, the exemplary modem 108 could be realized in various manners depending on the number of components implemented in software. The components most suited for either a software or a hardware implementation are the controller 302, the DPAL 304, the AMOS 306, the sample buffer management module 308, and the signal 25 processing tasks 312. Thus, although it is contemplated to implement other components in either hardware or software, the stated components are most commonly implemented in either hardware or software. Advantageously, implementation of the signal processing tasks 312 in software provides modularity and updating of individual tasks without affecting other components of the modem 108. In addition, implementing multiple individual signal

processing tasks 312 allows for more efficient usage of memory in the computer system 104 operating with the modem 108 according to the present invention.

Figure 4 is a block diagram of an exemplary computer communication system 400 for monitoring and controlling, in a computer system 401, a local modem 402 as it communicates 5 with a remote modem 404 of another computer system 405 over a telephone line 406. Similar to the computer communication system 100, the computer communication system 400 includes a modem monitor/control application 408 and a modem monitor/control interface 410. The local modem 402 is monitored/controlled just as the modem 108 is monitored and controlled. In addition, the remote modem 404 can be monitored by the 10 computer communication system 400 by using some of the bandwidth of the telephone line 406. Of course, if the communication devices were not modems and they communicated across something other than a telephone line, similar usage of the bandwidth on the line would enable functionality of the communication system 400.

A data stream is created on the telephone line 406 between the local modem 402 and 15 the remote modem 404, the data stream representing a modem connection. The telephone line 406 is used to transfer modem diagnostics and/or control information to/from the remote modem 404 by either "stealing" some of the data bits or using an alternative channel whenever applicable (e.g., V.34 control channel). The extraction of the diagnostic information can be performed in one of at least two manners.

- 20 1. A specific application can be run on the remote side that extracts modem parameters from the data stream and then sends them via the modem to the local side. The specific application can also receive control commands from the local modem and apply the commands to the remote modem.
- 25 2. The remote modem itself multiplexes the diagnostic information in the data stream (or the control channel) and monitors control commands without any interference from outside. The multiplexing/demultiplexing can be performed on any of the following two levels: by a data pump, or by an active data protocol (V.42, V.17). This second implementation for extracting diagnostic information from the data

stream is particularly suitable for software modem implementations where the modem can be easily modified for that kind of data manipulation and a wide variety of modem parameters can be extracted (e.g., see ModemMonCtrl API of the Appendixes).

5        In this manner, modem parameters from the remote modem 404 can be monitored and the remote modem 404 can be controlled with new parameters being set in the remote modem 404 from the computer communication system 400. Of course, the data stream between the local modem 402 and the remote modem 404 is ongoing and, potentially, the data stream passes without interruption from the computer communication system 400 regardless of  
10      whether the modems are software, hardware, or combination software/hardware modems.

Figure 5 is a block diagram of exemplary computer communication systems operating modem monitor/control applications, respectively, on both a client modem 500 in a local computer system 501 and a server modem 502 in a remote computer system 503. The local and remote computer systems 501, 503 communicate across a peer-to-peer network 504. A  
15      client computer communication system 506 communicates with the client modem 500 while telecommunication software or application 508 having an operating system communication driver 510 uses the client modem 500 to maintain a modem connection across the peer-to-peer network 504. Similar to the computer communication systems 100 and 400, the client computer communication system 506 operates in a manner to monitor/control the client  
20      modem 500 and/or the server modem 502. The difference in this embodiment pertains to the computer communication systems including both the client computer communication system 506 and a server computer communication system 512. This arrangement is provided to ensure accurate monitoring and/or controlling of both server and client modems. In addition, this embodiment demonstrates the flexibility of the system according to the present invention  
25      and expands the control options of the software.

Figure 6 is a block diagram of exemplary computer communication systems operating modem monitor/control applications, respectively, on both a local computer system 600 having a local modem 601 and a remote computer system 602. The local and remote computer systems 600, 602 communicate across a network 604. This embodiment illustrates

a structure similar to Figure 5 except that, rather than peer-to-peer network 504, the local and remote computer systems 600, 602 communicate across the network 604, the network 604 often being the Internet. Of course, the same advantages and benefits previously described in relation to modem monitoring, controlling, and configuring (a.k.a., diagnostics) are realized 5 when the modem 601 operates so as to access the Internet through an Internet service providers (ISP). Of course, if a communication device other than modem 601 is used to implement communication across the network 604, monitoring/controlling/configuring can be performed in a similar manner as described herein.

Figure 7 is a block diagram of an exemplary computer communication system 10 operating according to simple network management protocol (SNMP) parameters such that computer systems 700 can perform remote trouble shooting of a modem 702 in another computer system 704 by viewing a modem web page 706. This exemplary embodiment demonstrates how a single manager or system administrator, SNMP server 708, monitors and controls numerous client modems across a network 710. The network 710 is commonly a 15 network such as the Internet. In this embodiment, SNMP, a common network management protocol, serves as the underlying communication protocol for the computer systems 700, 704. Thus, a single manager, SNMP server 708, can monitor and control modems such as the modem 702. Of course, other network management protocols could be used to implement the principles according to the present invention and the description of SNMP operating over the 20 network 710 should not be construed to limit the appended claims.

The above-listed sections and included information are not exhaustive and are only exemplary for certain computer/modem/network systems. The particular sections and included information in a particular embodiment may depend upon the particular implementation and the included devices and resources. Although a system and method 25 according to the present invention has been described in connection with the preferred embodiments, it is not intended to be limited to the specific form set forth herein, but, on the contrary, it is intended to cover such alternatives, modifications, and equivalents as can be reasonably included within the spirit and scope of the invention as defined by the appended claims.

## Appendix A

```
#ifndef _MODEM_CTRL_H_
#define _MODEM_CTRL_H_

5  #include <Windows.h>      // To provide types definition, can be replaced by
any alternative type defining file
#include "ModemCodes.h"

10 #ifdef __cplusplus
extern "C" {
#endif

15     VOID WINAPI ModemGetLastError( PCHAR pBuf, DWORD nBuf );

20     /*
The GetModemCodesVersion function returns the version of the control codes
header file.
It should be used to verify coherence between the modem control API user
and provider.
*/
25     DWORD WINAPI ModemGetCodesVersion();

30     /*
The ModemOpen function returns a handle that can be used to access
a data-pump object.

Parameters:
dwDpIdCode - Specifies the type identification code of the data pump.
This value identifies the specific data pump to be monitored or
controlled.
The data pump type identification codes are defined by the type
RK_DP_IDS
35     (file "ModemCodes.h").

Return Values:
If the specified data pump type exists and the function succeeds,
the return value is an open handle to the specified modem.
40 If the function fails, the return value is INVALID_HANDLE_VALUE.
*/
45     HANDLE WINAPI ModemOpen(
        DWORD dwDpIdCode
);

50     /*
The ModemClose function closes an open object handle.

Parameters:
hModem - Identifies an open object handle to one of the following objects:
```

## CModem

## Return Values:

5 If the function succeeds, the return value is TRUE.  
If the function fails, the return value is FALSE.  
\*/

10 BOOL WINAPI ModemClose(  
 HANDLE hModem // handle to object to close  
);

/\*

15 The functions: ModemConfigure, ModemControl, ModemMonitor  
send a control code to a specified CModem object,  
causing the corresponding device to perform the specified operation.  
ModemConfigure has to be called BEFORE the specified modem has been  
activated.

20 ModemControl and ModemMonitor may be called DURING modem operation.

## Parameters:

hModem - Handle to the CModem instance that is to perform the operation.  
Call the CreateModem function to obtain a CModem handle.

25 dwConfigCode/dwControlCode/dwMonitorCode - Specify the control code for the  
operation.

This value identifies the specific configuration to be  
performed by

ModemConfigure/ModemControl/ModemMonitor respectively.

30 The control codes are defined by types  
RK\_CFG\_CODES/RK\_CTL\_CODES/RK\_MON\_CODES

(file "ModemCodes.h").

pInBuffer - Pointer to a buffer that contains the data required to perform  
the operation.

35 This parameter can be NULL if the dwConfigCode parameter  
specifies an operation

that does not require input data.

nInBufferSize - Specifies the size, in bytes, of the buffer pointed to by  
pInBuffer.

40 pOutBuffer - Pointer to a buffer that receives the operation's output data.  
This parameter can be NULL if the dwConfigCode parameter

specifies an operation

that does not produce output data.

nOutBufferSize - Specifies the size, in bytes, of the buffer pointed to by  
pOutBuffer.

45 pBytesReturned - Pointer to a variable that receives the size, in bytes,  
of the data stored into the buffer pointed to by pOutBuffer.

## Return Values:

50 If the function succeeds, the return value is TRUE.  
If the function fails or the specified operation is not supported  
for the specified object, the return value is FALSE.  
\*/

55 BOOL WINAPI ModemConfigure(

```
        HANDLE hModem,           // handle to CModem instance of
interest
        DWORD dwConfigCode,    // control code of operation to perform
        PVOID pInBuffer,       // pointer to buffer to supply input data
5        DWORD nInBufferSize,  // size of input buffer
        PVOID pOutBuffer,      // pointer to buffer to receive output data
        DWORD nOutBufferSize,  // size of output buffer
        PDWORD pBytesReturned // pointer to variable to receive output byte
count
10    );

        BOOL WINAPI ModemControl(
            HANDLE hModem,           // handle to CModem instance of
interest
            DWORD dwControlCode,    // control code of operation to perform
            PVOID pInBuffer,       // pointer to buffer to supply input data
15            DWORD nInBufferSize,  // size of input buffer
            PVOID pOutBuffer,      // pointer to buffer to receive output data
            DWORD nOutBufferSize,  // size of output buffer
            PDWORD pBytesReturned // pointer to variable to receive output byte
20        count
            );
count
        );

        BOOL WINAPI ModemMonitor(
            HANDLE hModem,           // handle to CModem instance of
interest
            DWORD dwMonitorCode,    // control code of operation to perform
            PVOID pInBuffer,       // pointer to buffer to supply input data
            DWORD nInBufferSize,  // size of input buffer
30            PVOID pOutBuffer,      // pointer to buffer to receive output data
            DWORD nOutBufferSize,  // size of output buffer
            PDWORD pBytesReturned // pointer to variable to receive output byte
count
            );
35

#ifdef __cplusplus
}
#endif
40#endif // _MODEM_CTRL_H_
```

## Appendix B

```

#ifndef _MODEM_CODES_H_
#define _MODEM_CODES_H_

5  #define MODEM_CODES_VERSION           8

    // rate masks returned by RKMON_SUPPORTED_BIT_RATE
    #define RK_RATE_MASK_75                0x00000001
    #define RK_RATE_MASK_300               0x00000002
10  #define RK_RATE_MASK_600               0x00000004
    #define RK_RATE_MASK_1200              0x00000008
    #define RK_RATE_MASK_2400              0x00000010
    #define RK_RATE_MASK_4800              0x00000020
    #define RK_RATE_MASK_7200              0x00000040
15  #define RK_RATE_MASK_9600              0x00000080
    #define RK_RATE_MASK_12000             0x00000100
    #define RK_RATE_MASK_14400             0x00000200
    #define RK_RATE_MASK_16800             0x00000400
    #define RK_RATE_MASK_19200             0x00000800
20  #define RK_RATE_MASK_21600             0x00001000
    #define RK_RATE_MASK_24000             0x00002000
    #define RK_RATE_MASK_26400             0x00004000
    #define RK_RATE_MASK_28800             0x00008000
    #define RK_RATE_MASK_31200             0x00010000
25  #define RK_RATE_MASK_33600             0x00020000
    #define RK_RATE_MASK_32000             0x00040000
    #define RK_RATE_MASK_34000             0x00080000
    #define RK_RATE_MASK_36000             0x00100000
    #define RK_RATE_MASK_38000             0x00200000
30  #define RK_RATE_MASK_40000             0x00400000
    #define RK_RATE_MASK_42000             0x00800000
    #define RK_RATE_MASK_44000             0x01000000
    #define RK_RATE_MASK_46000             0x02000000
    #define RK_RATE_MASK_48000             0x04000000
35  #define RK_RATE_MASK_50000             0x08000000
    #define RK_RATE_MASK_52000             0x10000000
    #define RK_RATE_MASK_54000             0x20000000
    #define RK_RATE_MASK_56000             0x40000000
    // DataPump type codes
40  typedef enum {
        RKID_V32BIS = 0,
        RKID_V34,
        RKID_V22BIS,
45  RKID_V23,
        RKID_V21,
        RKID_V17,
        RKID_V29,
50  RKID_V27,
        RKID_V8,

```

```
        RKID_TONE_DET,  
        RKID_TONE_GEN,  
        RKID_DTMF_DET,  
5       RKID_DTMF_GEN,  
        RKID_CR_TONE_DET,  
        RKID_CR_TONE_GEN,  
  
        RKID_RKSAMPLE,  
10      RKID_ANS_DET,  
        RKID_ANS_GEN,  
        RKID_WINAC,  
        RKID_ROKV42,  
  
15      RKID_K56FLEX,  
        RKID_BELL103,  
        RKID_BELL212A,  
        RKID_SPKP,  
        RKID_VOICE,  
20      RKID_V90,  
  
        RKID_AMOS,  
  
25      RKID_LAST,  
} RK_DP_IDS;  
  
// Offset definitions:  
30      #define COMMON_RK_CODES      0  
  
#define RKSAMPLE_RK_CODES  2000  
#define WINAC_RK_CODES   3000  
#define V42_RK_CODES    4000  
35      #define AUTOMODE_RK_CODES  6000  
#define V8_RK_CODES     7000          // V8, V8BIS  
  
#define V21_RK_CODES    10000  
40      #define V22_RK_CODES    11000          // V22, Bell-212A  
#define FSK_RK_CODES   12000          // V23, Bell-103  
  
#define FAX_RK_CODES   14000          // V27, V27BIS, V27TER, V29, V17  
45      #define V32_RK_CODES   16000          // V32, V32BIS  
  
#define V34_RK_CODES   18000  
  
#define V90_RK_CODES   20000          // K56FLEX, V90  
50      #define SPKP_RK_CODES  25000  
  
#define VOICE_RK_CODES 26000  
55      #define AMOS_RK_CODES  27000
```

```

// Modem Config Codes
(In)      Parameter (Out)          Parameter
5
typedef enum
{
// ***** Common Constants *****

    // Select Symbol Rate (no impact if Autorate is enabled)
    RKCFG_TX_SYMBOL_RATE = COMMON_RK_CODES, //          INT -
10  Symbol Rate None
        RKCFG_RX_SYMBOL_RATE, //          INT
    - Symbol Rate None
        // Force Bit Rate
        RKCFG_BIT_RATE_RX_MAX, //          INT
15  - Bit Rate None
        RKCFG_BIT_RATE_TX_MAX, //          INT
    - Bit Rate None
        RKCFG_BIT_RATE_RX_MIN, //          INT
    - Bit Rate None
        RKCFG_BIT_RATE_TX_MIN, //          INT
20  - Bit Rate None

    // Select connection type ( Half or Full Duplex )
    RKCFG_CONNECTION_TYPE, //          DWORD
25  (FDplex=0,HDplex=1) None

    // Tx Transmittion Power: {Minimum, Maximum, Default, Offset}
    // (values in dBm, offset in dB). Offset is for compensation on
hardware gain.
30
    RKCFG_TX_SIGNAL_POWER, //          INT
    char[4] None

    // Enable/Disable Rate Renegotiation
    RKCFG_RENEG_ENABLE, //          INT
35  BOOL - Yes/No None
    // Enable/Disable Retrain
    RKCFG_RETRAIN_ENABLE, //          INT
    BOOL - Yes/No None

40
    // Enable/Disable Rx Freeze
    RKCFG_RX_FREEZE_ENABLE, //          INT
    BOOL - Yes/No None
    // Enable/Disable Echo Canceller Freeze
    RKCFG_EC_FREEZE_ENABLE, //          INT
45  BOOL - Yes/No None

    RKCFG_RECORD_SESSION, //          INT
    BOOL - Yes/No None
    RKCFG_SESSION_NAME, //          INT
50  char * name None

    RKCFG_NO_CARRIER_TIMEOUT, //          INT
    DWORD - in seconds None

```

```
    RKCFG_START_AT_DATA, //  
    BOOL - Yes/No      None  
  
    RKCFG_REMOTE_IS_ROCKWELL, //  
5     BOOL - Yes/No      None  
  
    RKCFG_MODEM_SETTINGS,  
// ***** Win AC Constants *****  
  
10    RKCFG_EC_MODE = WINAC_RK_CODES, //  
        DWORD(ERROR_CONTROL_MODE)  
        RKCFG_CMPRS_MODE, //  
        DWORD(COMPRESSION_MODE)  
        RKCFG_ACTIVE_MODULATION, //  
15    DWORD(RK_DP_IDS)  
  
// ***** Auto-Mode Constants *****  
  
    // Enable/Disable Automode  
20    RKCFG_AUTOMODE_ENABLE =AUTOMODE_RK_CODES, //  
        BOOL -  
        Yes/No      None  
  
    // Transmit Timeout for detection for V32  
    RKCFG_TRANSMIT_TIMEOUT, //  
25    DWORD ms      None  
  
// ***** V8 Constants *****  
  
30    RKCFG_V8_SUPPORT_CI = V8_RK_CODES, //  
        BOOL -  
        Yes/No      None  
  
    RKCFG_V8_CI_CALLING_FUNCTION_SEQUENCE, //  
        BYTE  
        None  
  
35    RKCFG_V8_CI_ON_CADENCE, //  
        DWORD ms cadence None  
        RKCFG_V8_CI_OFF_CADENCE, //  
        DWORD ms cadence None  
40    RKCFG_V8_AS_CI_DET, //  
        BOOL      None  
// ***** V21 Constants *****  
  
    RKCFG_V21RX_HIGH_CHANNEL = V21_RK_CODES, //  
45    BOOL -  
    Yes/No      None  
    RKCFG_V21TX_HIGH_CHANNEL, //  
    BOOL - Yes/No      None  
  
    RKCFG_V21_DATA_MODE, //  
50    BOOL      None  
  
// ***** V22 Constants (V22, Bell-212A) *****  
  
    RKCFG_V22_TO_BELL_212A = V22_RK_CODES, //  
55    BOOL -  
    Yes/No      None
```

```

// ***** FSK Modulations Constants (V23, Bell-103) *****
      RKCFG_FSK_BACK_CHANNEL = FSK_RK_CODES,      //      BOOL -
5      Yes/No      None

      RKCFG_FSK_V23_CHANNEL,                      //      BOOL - Yes/No      None

10     RKCFG_FSK_BELL103_CHANNEL,                //      BOOL - Yes/No      None

      RKCFG_FSK_FOR_CID,                         //      BOOL - Yes/No      None
15     // ***** Fax Constants (V27, V29, V17) *****

      // Define Retrain between Pages as Short or Long
      RKCFG_LONG_RETRAIN = FAX_RK_CODES,          //      BOOL
      (TRUE=Long) None

20     // ***** V32 Constants *****

      RKCFG_V32BIS_TO_V32 = V32_RK_CODES,          //      BOOL -
      Yes/No      None

25     RKCFG_V32_TRELLIS_SUPPORT,                //      BOOL - Yes/No      None

      // ***** V34 Constants *****

30     // Select Carrier Frequency
      RKCFG_RX_CARRIER_FREQ = V34_RK_CODES,          //      V34_carrier_t      None

      // Enable/Disable Transmit Power Drop
35     RKCFG_TX_POWER_DROP_ENABLE,                //      BOOL - Yes/No      None
      // Select Transmit Power Level
      RKCFG_TX_POWER_DROP,                         //      INT
      - Level      None
      // Select Requested Power Drop
40     RKCFG_REQUESTED_POWER_DROP,                //      DWORD      None

      // Enable/Disable Precoding
45     RKCFG_PRECODING_ENABLE,                  //      BOOL - Yes/No      None
      // Set Precoding Coefficients
      RKCFG_PRECODING_COEFFS,                     //      SHORT[6] -
      Array of coeffs      None

50     // Transmitter Preemphasis Filter
      RKCFG_TX_PREEMPHASIS_FILTER,                //      INT -
      Filter Index      None
      // Requested Preemphasis Filter

```

```

        RKCFG_REQUESTED_PREEMPHASIS_FILTER,           //           INT -
      Filter Index      None

      // Enable/Disable Constellation Expansion
5       RKCFG_CONSTELATION_EXPAND_ENABLE,           //           BOOL -
      Yes/No        None
      // Enable/Disable Warping
      RKCFG_WARP_ENABLE,                          //           BOOL -
      BOOL - Yes/No      None
10      // ***** V90 Constants (K56FLEX, V90) *****

      // set the encoding law for flex 1 indicates A-law coding, 0 indicates
      u-law
      RKCFG_ENCODING_LAW = V90_RK_CODES,           //           BOOL
15      (TRUE=A_Law)      None

      // ***** SpeakerPhone Constants *****

      // Hardware Delay
20      RKCFG_EC_DELAY = SPKP_RK_CODES,           // {SPKP_MODULE, INT - No of
      Samples}      None
      // Cross-Correlator Length
      RKCFG_CC_LENGTH,                          //           INT
      - No of Taps      None
25      RKCFG_DMP_MASK,

      RKCFG_INITIAL_FULL_DUPLEX_MEASURE,

30      } RK_CFG_CODES;

      // Modem Control Codes
      typedef enum
      {
35      // ***** Common Constants *****

      // Initiate Retrain
      RKCTL_RETRAIN = COMMON_RK_CODES,           //           None
      None
40      // Initiate Rate Renegotiation
      RKCTL_RENEG,                           //           None
      INT - Bit Rate      None
      // Terminate Connection Gracefully
      RKCTL_CLEARDOWN,                      //           None
45      None      None

      // Squelch Tx Signal
      RKCTL_TX_SQUELCH,                      //           None
      None      None
50      // Use the SendCommand
      RKCTL_SEND_COMMAND,                   //           None
      {DWORD[2]- Command, Param}      None

55      // WinAC constants

```

```

        RKCTL_MODEM_SLEEP = WINAC_RK_CODES,           //          DWORD
        None

// ***** Fax Constants (V27, V29, V17) *****
5
        // Define Retrain between Pages as Short or Long
        RKCTL_LONG_RETRAIN = FAX_RK_CODES,           //          BOOL
        (TRUE=Long) None

10 // ***** V34 Constants *****

        // Must be sent before RKMON_DATA_RES_ECHO_GET
        RKCTL_DATA_RES_ECHO_REQUEST=V34_RK_CODES, //          None
        None

15 // ***** SpeakerPhone Constants *****

        // Speakerphone Mode (FD, HD, HS)
        RKCTL_SPKP_MODE = SPKP_RK_CODES,           //          SPKPMode
        None

20        // Output Mute
        RKCTL_IO_MUTE,                           //          None
        {SPKP_PROBE,BOOL - Yes/No}      None
        // Echo Cancellers
25        RKCTL_FILTER_LENGTH,                  // {SPKP_MODULE, INT - No
        of Taps}      None
        RKCTL_EC_OPERATE,                     // {SPKP_MODULE, BOOL -
        Yes/No}      None
        RKCTL_ADAPT_ENABLED,                  // {SPKP_MODULE, BOOL -
30        Yes/No}      None
        // AGC and Sw-Loss
        RKCTL_AMP_ENABLED,                   //          None
        {SPKP_MODULE, BOOL - Yes/No}
        // Gains
35        RKCTL_GAIN,                      // {SPKP_MODULE*, INT*/FLOAT* -
        Gain,GAIN_FORMAT*}      None

        RKCTL_INIT_GAIN,
        RKCTL_MAX_GAIN,
40        RKCTL_FULL_DUPLEX_MEASURE,

        RKCTL_NOISE_INSERTION_LENGTH,
        RKCTL_NOISE_INSERTION_ENABLE,

45        RKCTL_FADE_IN_LENGTH,
        RKCTL_FADE_IN_ENABLE,

        RKCTL_UPSTEP,

50        RKCTL_MIN_LINE_OUT_POWER,
        RKCTL_LINE_OUT_SILENCE_GAIN_REDUCTION,
        RKCTL_CREATE_DATAPUMP = AMOS_RK_CODES,

```

```
RKCTL_DESTROY_DATAPUMP,  
 } RK_CTL_CODES;  
  
5 // Modem Monitor Codes  
typedef enum  
{  
// ***** Common Constants *****  
  
10 RKMON_TX_SAMPLE_RATE = COMMON_RK_CODES, // None  
    DWORD - Sample Rate  
    RKMON_RX_SAMPLE_RATE, //  
    None DWORD - Sample Rate  
    RKMON_TX_SYMBOL_RATE, //  
    None INT - Symbol Rate  
    RKMON_RX_SYMBOL_RATE, //  
    None INT - Symbol Rate  
    RKMON_TX_BIT_RATE, //  
    None INT - Bit Rate  
    RKMON_RX_BIT_RATE, //  
    None INT - Bit Rate  
  
    RKMON_TX_CARRIER_FREQUENCY, // None  
    DWORD - (Hz)  
    RKMON_RX_CARRIER_FREQUENCY, // None  
    DWORD - (Hz)  
    RKMON_RX_SIGNAL_POWER, //  
    None Float - (dBm)  
    RKMON_RX_SIGNAL_POWER, //  
    None Float - (dBm)  
  
    // Constellation points  
    RKMON_RX_SCATTER, //  
    None float* - pointer to pairs of points  
    // Gain needed for scatter plot  
    RKMON_RX_NORM_FACTOR, //  
    None float  
  
    RKMON_ROUND_TRIP_DELAY, //  
    None INT - R.T.D in 8k samples per sec.  
  
    // M.S.E at Rate selection [dB]  
    RKMON_BASE_MSE, //  
    None Float  
    // Mean Square Error [dB]  
    RKMON_MSE, //  
    None Float  
  
    // Signal to Noise Ratio (dB)  
    RKMON_SNR, //  
    None Float  
    RKMON_EQM, //  
    None float - (dB)
```

```

        RKMON_SUPPORTED_BIT_RATES_MASK ,           //           None
        DWORD (masks of RK_RATE_MASK_ defined above)
        RKMON_FE_ECHO_DELAY ,                     //

5       RKMON_AUDIO_TX_SAMPLE_RATE,           //           None
        None          DWORD - Sample Rate
        RKMON_AUDIO_RX_SAMPLE_RATE,           //           None
        None          DWORD - Sample Rate

10      RKMON_SETTINGS_INFO,
        RKMON_SETTINGS_BLOCKS,
// ***** Rksample Constants *****

        // Num of microseconds in last interrupt
15      RKMON_LAST_INT_CPU = RKSAMPLE_RK_CODES, //           None
        DWORD
        // Num of microseconds between last 2 interrupts
        RKMON_LAST_INT_LATENCY ,           //           None
        None          DWORD
20      // Num of microseconds in longest interrupt
        RKMON_MAX_INT_CPU ,           //           None
        None          DWORD
        // Longest latency between 2 interrupts (microseconds)
        RKMON_MAX_INT_LATENCY ,           //           None
        None          DWORD
        // Num of samples overrun occurred in the past
        RKMON_SAMPLES_OVERRUNS ,           //           None
        None          DWORD
        // Num of samples occurred in the past
30      RKMON_SAMPLES_UNDERRUNS,           //           None
        None          DWORD
        // Num of bus overruns occurred in the past
        RKMON_BUS_OVERRUNS ,           //           None
        None          DWORD
        // Num of bus underruns occurred in the past
35      RKMON_BUS_UNDERRUNS,           //           None
        None          DWORD
        // Operating speed
        RKMON_OPERATING_SPEED,           //           None
        None          DWORD

40      // ***** WinAc Constants *****

        // Index (WinAc style) of the active modulation
45      RKMON_ACTIVE_MODULATION=WINAC_RK_CODES, //           None
        DWORD
        RKMON_MODEM_STATE,           //           None
        None          DWORD
        RKMON_MODEM_SLEEP,           //           None
        None          DWORD
        // RKMON_CALL_SETUP_RES - identical
        // to field no. 1 in AT#UD
        RKMON_CALL_SETUP_RES,           //           None
        None          DWORD
        // RKMON_MULTI_MEDIA_MODE - identical
55      //           None

```

```
// to field no. 2 in AT#UD
RKMON_MULTI_MEDIA_MODE, //  
5 None DWORD
// RKMON_V8_CM - identical to field no.
// 4 in AT#UD. Returns a pointer to string.
RKMON_V8_CM, //  
None PCHAR
// RKMON_V8_JM - identical to field no.
// 5 in AT#UD. Returns a pointer to string.
RKMON_V8_JM, //  
10 None PCHAR
// RKMON_TX_NEG_RES - identical to
// field no. 20 in AT#UD
RKMON_TX_NEG_RES, //  
15 None DWORD
// RKMON_RX_NEG_RES - identical to
// field no. 21 in AT#UD
RKMON_RX_NEG_RES, //  
20 None DWORD
// RKMON_CARRIER_LOSS_EV_CNT -
// identical to field no. 30 in AT#UD
RKMON_CARRIER_LOSS_EV_CNT, //  
25 None DWORD
// RKMON_RATE_RENEG_EV_CNT -
// identical to field no. 31 in AT#UD
RKMON_RATE_RENEG_EV_CNT, //  
30 None DWORD
// RKMON_RTRN_REQ - identical to field
// no. 32 in AT#UD
RKMON_RTRN_REQ, //  
35 None DWORD
// RKMON_RTRN_GRANTED - identical to
// field no. 33 in AT#UD
RKMON_RTRN_GRANTED, //  
40 None DWORD
// RKMON_PROTOCOL_NEG_RES - identical
// to field no. 40 in AT#UD
RKMON_PROTOCOL_NEG_RES, //  
45 None DWORD
// RKMON_EC_FRAME_SIZE - identical to
// field no. 41 in AT#UD
RKMON_EC_FRAME_SIZE, //  
50 None DWORD
// RKMON_EC_LINK_TIMEOUTS - identical
// to field no. 42 in AT#UD
RKMON_EC_LINK_TIMEOUTS, //  
55 None DWORD
// RKMON_CMPRS_NEG_RES - identical to
// field no. 44 in AT#UD
RKMON_CMPRS_NEG_RES, //
```

```

// RKMON_CMPRS_DICT_SIZE - identical to
// field no. 45 in AT#UD
RKMON_CMPRS_DICT_SIZE,                                // 5
None          DWORD
// RKMON_TX_FLOW_CTRL - identical to
// field no. 50 in AT#UD
RKMON_TX_FLOW_CTRL,                                // 10
None          DWORD
// RKMON_RX_FLOW_CTRL - identical to
// field no. 51 in AT#UD
RKMON_RX_FLOW_CTRL,                                // 15
None          DWORD
// RKMON_TOTAL_TX_CHARS - identical to
// field no. 52 in AT#UD
RKMON_TOTAL_TX_CHARS,                                // 20
None          DWORD
// RKMON_TOTAL_RX_CHARS - identical to
// field no. 53 in AT#UD
RKMON_TOTAL_RX_CHARS,                                // 25
None          DWORD
// RKMON_TERMINATION_CAUSE - identical
// to field no. 60 in AT#UD
RKMON_TERMINATION_CAUSE,                                // 30
None          DWORD
// RKMON_CALL_WAIT_EV_CNT - identical
// to field no. 61 in AT#UD (not supported)
RKMON_CALL_WAIT_EV_CNT,                                // 35
None          DWORD
RKMON_CPU_VENDOR,          PCHAR
None          DWORD
RKMON_CACHE_SIZE,          PCHAR
None          DWORD
RKMON_NUMBER_CALLED,          PCHAR
RKMON_TIMER_RESOLUTION,          DWORD
                                            //          None
// ***** V42 Constants *****
40      // Number of V42 BLERS
RKMON_BLER      = V42_RK_CODES,          //          None
                                            DWORD
// ***** Fax Constants (V27, V29, V17) *****
45      // Whether Retrain between Pages is Short or Long
RKMON_LONG_RETRAIN = FAX_RK_CODES,          //          None
                                            BOOL (TRUE=Long)
50      // ***** V34 Constants *****
// Transmit Power Drop [dB]
RKMON_TX_POWER_DROP = V34_RK_CODES,          //          None
                                            INT
55      // Power Drop [dB] that was requested from remote modem

```

```

    RKMON_RX_POWER_DROP,                                //  

    None          INT  

5     // Transmitter Preemphasis Filter  

    RKMON_TX_PREEMPHASIS_FILTER,                      //  

        INT - Filter Index  

    // other side's Preemphasis Filter  

    RKMON_RX_PREEMPHASIS_FILTER,                      //  

        INT - Filter Index  

10
  

    // Residual Echo in training [dB]  

    RKMON_TRN_RESIDUAL_ECHO,                          //  

    None          Float  

15     // Residual Echo in data [dB]  (must be sent after  

    RKCTL_DATA_RES_ECHO_REQUEST)  

    RKMON_DATA_RES_ECHO_GET,                          //  

    None          Float  

    // Near End Echo [dB]  

20     RKMON_NE_ECHO_POWER,                          //  

    None          Float  

    // Far End Echo [dB]  

    RKMON_FE_ECHO_POWER,                          //  

    None          Float  

25
  

    // Timing Drift [ppm]  

    RKMON_TIMING_DRIFT,                            //  

    None          Float  

    // Frequency Offset [Hz]  

30     RKMON_FREQ_OFFSET,                          //  

    None          Float  

// ***** V90 Constants (K56FLEX, V90) *****  

35     // Robbed Bits Signaling  

    RKMON_RBS_DETECTED = V90_RK_CODES,           //          None  

        DWORD RBS frame 0 to 63 (1' indicate robbed bit)  

    // PCM Pad  

    RKMON_PAD_DETECTED,                          //  

40     None          DWORD PAD 0=NORMAL ,3=3dBPad 6=6dBPad  

    // High Pass filter enabled  

    RKMON_HIGHPASS_FILTER_ENABLED,                //  

    None          BOOL - Yes/No  

45
  

// ***** SpeakerPhone Constants *****  

    // Speakerphone Mode (FD, HD, HS)  

    RKMON_SPKP_MODE = SPKP_RK_CODES,           //          None  

50     SPKPMode  

    // State  

    RKMON_STATE,                                //  

    None          SPKPState  

    // Input-Output Mute

```

	RKMON_IO_MUTE,	//
	SPKP_PROBE	BOOL - Yes/No
	RKMON_SATURATION,	//
	SPKP_PROBE	BOOL - Yes/No
5	RKMON_DC_LEVEL,	//
	SPKP_PROBE	FLOAT
	// Echo Cancellers	
	RKMON_FILTER_LENGTH,	//
	SPKP_MODULE	INT - No of Taps
10	RKMON_EC_OPERATE,	//
	SPKP_MODULE	BOOL - Yes/No
	RKMON_ADAPT_ENABLED,	//
	SPKP_MODULE	BOOL - Yes/No
	RKMON_EC_DELAY,	//
15	SPKP_MODULE	INT - No of Samples
	// AGC and Sw-Loss	
	RKMON_AMP_ENABLED,	//
	SPKP_MODULE	BOOL - Yes/No
	// Powers	
20	RKMON_POWER,	//
	SPKP_PROBE	FLOAT - Power [dB]
	RKMON_NOISE_POWER,	//
	SPKP_PROBE	FLOAT - Power [dB]
	// Gains	
25	RKMON_GAIN,	//
	{SPKP_MODULE,GAIN_FORMAT}	INT/FLOAT - Gain [Scaled,dB,Linear]
	// Gain Estimations	
	RKMON_ECHO_PATH_GAIN,	//
	ECHO_PATH	FLOAT - Gain [dB]
30	RKMON_EC_GAIN,	//
	SPKP_MODULE	FLOAT - Gain [dB]
	RKMON_RES_ECHO_GAIN,	//
	SPKP_MODULE	FLOAT - Gain [dB]
35	RKMON_INIT_GAIN,	
	RKMON_MAX_GAIN,	
	RKMON_FULL_DUPLEX_MEASURE,	
	RKMON_TONE_DETECT,	
40	RKMON_NOISE_INSERTION_LENGTH,	
	RKMON_NOISE_INSERTION_ENABLE,	
	RKMON_FADE_IN_LENGTH,	
45	RKMON_FADE_IN_ENABLE,	
	RKMON_UPSTEP,	
	RKMON_MIN_LINE_OUT_POWER,	
50	RKMON_DMP_MASK,	
	RKMON_LINE_OUT_SILENCE_GAIN_REDUCTION,	
55	RKMON_INITIAL_FULL_DUPLEX_MEASURE,	

```
// ***** Voice Constants *****
    RKMON_VOICE_AVG_POWER = VOICE_RK_CODES,
5 } RK_MON_CODES;

// SPKP Modules
typedef enum {
    LINEIN_AMP,
10   LEC,    TONE_DET,   RX_SD,   RX_SW_LOSS,   RX_AGC,
    SPKR_AMP,
    MIC_AMP,
    AEC,    TX_SD,    TX_SW_LOSS,   TX_AGC,
    LINEOUT_AMP,
15   ALL_MODULES
} SPKP_MODULE;

// SPKP Probing points
typedef enum {
20   LINEIN,
    LEC_IN,  LEC_OUT,  RX_AGC_OUT,
    SPKR,
    MIC,
    AEC_IN,  AEC_OUT,  TX_AGC_OUT,
25   LINEOUT,
    ALL_PROBES
} SPKP_PROBE;

// Gain Format: dB or Scaled 0-255
30 typedef enum { SCALED, DB, LINEAR } GAIN_FORMAT;

// Echo Path
typedef enum { ACOUSTIC, LINE } ECHO_PATH;

35 // Error Control Mode
typedef enum { EC_FORCED, EC_OFF, EC_ON } ERROR_CONTROL_MODE;

// Modem global state
typedef enum { STATE_INITIALIZING, STATE_IDLE, STATE_ORIGINATE,
40 STATE_ANSWER,
45 STATE_V8BIS_HS, /* STATE_MST, */ STATE_TRAINING,
    STATE_CONNECTED,
    STATE_ESCAPED, STATE_LAL, STATE_LAL_ESCAPED,
    STATE_RDL } MODEM_STATE;

// Compression Mode
typedef enum { CMPRS_OFF, CMPRS_ON } COMPRESSION_MODE;

#endif // _MODEM_CODES_H_
```

## Appendix C

```

#include "dlldefs.h"
#include "ModemCtrl.h"
#include "appinterface.h"

5      #define MAX_ERRORMSG_LEN      200

10     HANDLE          hModCtrlVxd = NULL;
     char      ErrorMsg[MAX_ERRORMSG_LEN];

HANDLE WINAPI ModemOpen( DWORD Code )
{
    PCLIENT_INFO      pClient;

15    if ( hModCtrlVxd == NULL || hModCtrlVxd == INVALID_HANDLE_VALUE ) {
#ifndef WINDOWS_NT
        hModCtrlVxd = CreateFile( "\\\.\MODCTRL.VXD", 0, 0, NULL,
                                0, FILE_FLAG_DELETE_ON_CLOSE,
20        NULL );
#else
        hModCtrlVxd = CreateFile( "\\\.\MODCTRL0",
                                GENERIC_READ | GENERIC_WRITE,
                                FILE_SHARE_READ,
                                NULL,
                                OPEN_EXISTING,
                                0,
                                NULL );
25    #endif
30    if ( hModCtrlVxd == INVALID_HANDLE_VALUE ) {
        strcpy( ErrorMsg, "Failed to load MODCTRL.VXD",
                MAX_ERRORMSG_LEN );
        return FALSE;
    }
35    }
     unsigned long      nBytes;

     BOOL rc = DeviceIoControl( hModCtrlVxd,
                                DP_OPEN_MODEM,
40                                &Code, sizeof(DWORD),
                                &pClient,
                                sizeof(PCLIENT_INFO),
                                &nBytes, NULL );
     if ( rc == 0 ) {
45    strcpy( ErrorMsg, "DeviceIoControl with Code DP_OPEN_MODEM
Failed",
                MAX_ERRORMSG_LEN );
     return NULL;
    }
50

     return (HANDLE)pClient;
}

```

```
}

BOOL WINAPI ModemClose( HANDLE hModem )
{
    5      if ( hModCtrlVxd == NULL ) {
            strncpy( ErrorMsg, "Can't close modem: ModCtrl.vxd not loaded",
                     MAX_ERRORMSG_LEN );
            return FALSE;
    }
    10     if ( hModem == NULL ) {
            strncpy( ErrorMsg, "Can't close modem: NULL handle",
                     MAX_ERRORMSG_LEN );
            return FALSE;
    }
    15     unsigned long      nBytes;
    PCLIENT_INFO      pClient = (PCLIENT_INFO)hModem;

    20     BOOL rc = DeviceIoControl( hModCtrlVxd,
                                    DP_CLOSE_MODEM,
                                    &pClient,
                                    sizeof(PCLIENT_INFO),
                                    NULL, 0 ,
                                    &nBytes, NULL );
    25     if ( rc == 0 ) {
            strncpy( ErrorMsg, "DeviceIoControl with Code DP_CLOSE_MODEM
Failed",
                     MAX_ERRORMSG_LEN );
            return NULL;
    }
    30     }
    return 1;
}

DWORD WINAPI ModemGetCodesVersion()
35     {
        return MODEM_CODES_VERSION;
    }

    40     BOOL WINAPI ModemConfigure(HANDLE hModem, DWORD dwConfigCode, PVOID
pInBuffer,
                                    DWORD nInBufferSize, PVOID pOutBuffer, DWORD
nOutBufferSize,
                                    PDWORD pBytesReturned )
    {
    45     BOOL
        MODEMCTRL_DATA
        PCLIENT_INFO
        DWORD BytesReturned;
    #ifdef WINDOWS_NT
        UPDATE_STRUCT
    50     UpdateClient;
    #endif

        if ( hModem == NULL ) {
            strncpy( ErrorMsg, "ModemConfigure failed: HANDLE is NULL",
                     MAX_ERRORMSG_LEN );
    }
```

```
        return FALSE;
    }

#define WINDOWS_NT
5     rc = DeviceIoControl( hModCtrlVxd,
                           DP_UPDATE_MODEM,
                           &hModem, sizeof(DWORD),
                           &UpdateClient, sizeof(UPDATE_STRUCT),
                           &BytesReturned, NULL );

10    if ( rc == FALSE )
    {
        return FALSE;
    }
    if (( UpdateClient.Status == DPACTIVE ) && (UpdateClient.ID !=
15    RKID_WINAC) ) {
#else
    if (( pClient -> Status == DPACTIVE ) && (pClient -> ID !=
    RKID_WINAC) ) {
#endif
        // Can't configure an active modulation, unless it is WinAC.
        strncpy( ErrorMsg, "Modem is active", MAX_ERRORMSG_LEN );
        return FALSE;
    }

25    #ifdef WINDOWS_NT
        ModemCtrlData.ObjectID = UpdateClient.ID;
    #else
        ModemCtrlData.ObjectID = pClient -> ID;
    #endif
30    ModemCtrlData.CodeIndex = dwConfigCode;
    ModemCtrlData.pInBuffer = pInBuffer;
    ModemCtrlData.cbInBuffer = nInBufferSize;
    ModemCtrlData.pOutBuffer = pOutBuffer;
    ModemCtrlData.cbOutBuffer = nOutBufferSize;
35    ModemCtrlData.pBytesReturned = pBytesReturned;

        rc = DeviceIoControl( hModCtrlVxd,
                           DP_CONFIGURE_MODEM,
                           &ModemCtrlData,
40    sizeof(MODEMCTRL_DATA),
                           NULL, 0,
                           &BytesReturned, NULL );

        if ( rc == FALSE )
45    strncpy( ErrorMsg, "DeviceIoControl with Code
    DP_CONFIGURE_MODEM Failed",
                           MAX_ERRORMSG_LEN );

        return rc;
50    }

    BOOL WINAPI ModemControl( HANDLE hModem, DWORD dwConfigCode, PVOID
pInBuffer,
                           DWORD nInBufferSize, PVOID pOutBuffer, DWORD
55    nOutBufferSize,
```

```
        PDWORD pBytesReturned )  
    {  
        BOOL rc;  
        PCLIENT_INFO pClient = (PCLIENT_INFO)hModem;  
        DWORD BytesReturned;  
5       #ifdef WINDOWS_NT  
        UPDATE_STRUCT UpdateClient;  
#endif  
        MODEMCTRL_DATA ModemCtrlData;  
10      if ( pClient == NULL ) {  
            strncpy( ErrorMsg, "ModemControl failed: HANDLE is NULL",  
MAX_ERRORMSG_LEN );  
            return FALSE;  
15      }  
#ifdef WINDOWS_NT  
        rc = DeviceIoControl( hModCtrlVxd,  
                               DP_UPDATE_MODEM,  
                               &hModem, sizeof(DWORD),  
20                               &UpdateClient, sizeof(UPDATE_STRUCT),  
                               &BytesReturned, NULL );  
        if ( rc == FALSE )  
        {  
            return FALSE;  
25        }  
        if ( UpdateClient.Status != DPACTIVE ) {  
#else  
        if ( pClient -> Status != DPACTIVE ) {  
#endif  
30            strncpy( ErrorMsg, "modem is not active", MAX_ERRORMSG_LEN );  
            return FALSE;  
        }  
  
#ifdef WINDOWS_NT  
35        ModemCtrlData.ObjectID = UpdateClient.ID;  
#else  
        ModemCtrlData.ObjectID = pClient -> ID;  
#endif  
        ModemCtrlData.CodeIndex = dwConfigCode;  
40        ModemCtrlData.pInBuffer = pInBuffer;  
        ModemCtrlData.cbInBuffer = nInBufferSize;  
        ModemCtrlData.pOutBuffer = pOutBuffer;  
        ModemCtrlData.cbOutBuffer = nOutBufferSize;  
        ModemCtrlData.pBytesReturned = pBytesReturned;  
45        rc = DeviceIoControl( hModCtrlVxd,  
                               DP_CONTROL_MODEM,  
                               &ModemCtrlData,  
sizeof(MODEMCTRL_DATA),  
50                               NULL, 0,  
                               &BytesReturned, NULL );  
  
        if ( rc == FALSE )  
            strncpy( ErrorMsg, "DeviceIoControl with Code DP_CONTROL_MODEM  
55 Failed",
```

```
        MAX_ERRORMSG_LEN );

    return rc;
}

5    BOOL WINAPI ModemMonitor( HANDLE hModem, DWORD dwConfigCode, PVOID
pInBuffer,
                               DWORD nInBufferSize, PVOID pOutBuffer, DWORD
nOutBufferSize,
10                           PDWORD pBytesReturned )

{
    BOOL
    PCLIENT_INFO
    MODEMCTRL_DATA
15    DWORD BytesReturned;
    #ifdef WINDOWS_NT
        UPDATE_STRUCT
        UpdateClient;
    #endif

20    if ( pClient == NULL ) {
        strncpy( ErrorMsg, "ModemMonitor failed: HANDLE is NULL",
MAX_ERRORMSG_LEN );
        return FALSE;
    }

25    #ifdef WINDOWS_NT
        rc = DeviceIoControl( hModCtrlVxd,
                               DP_UPDATE_MODEM,
                               &hModem, sizeof(DWORD),
30                               &UpdateClient, sizeof(UPDATE_STRUCT),
                               &BytesReturned, NULL );
        if ( rc == FALSE )
        {
            return FALSE;
        }
        if ( UpdateClient.Status != DPACTIVE ) {
35        #else
            if ( pClient -> Status != DPACTIVE ) {
        #endif
40            //strncpy( ErrorMsg, "Modem is not active", MAX_ERRORMSG_LEN );
            return FALSE;
        }

45    #ifdef WINDOWS_NT
        ModemCtrlData.ObjectID = UpdateClient.ID;
    #else
        ModemCtrlData.ObjectID = pClient -> ID;
    #endif
50    ModemCtrlData.CodeIndex = dwConfigCode;
    ModemCtrlData.pInBuffer = pInBuffer;
    ModemCtrlData.cbInBuffer = nInBufferSize;
    ModemCtrlData.pOutBuffer = pOutBuffer;
    ModemCtrlData.cbOutBuffer = nOutBufferSize;
55    ModemCtrlData.pBytesReturned = pBytesReturned;
```

```
    rc = DeviceIoControl( hModCtrlVxd,
                          DP_MONITOR_MODEM,
                          &ModemCtrlData,
5    sizeof(MODEMCTRL_DATA),
                          pOutBuffer, nOutBufferSize,
                          pBytesReturned, NULL );

    if ( rc == FALSE )
10      strncpy( ErrorMsg, "DeviceIoControl with Code DP_MONITOR_MODEM
Failed",
                  MAX_ERRORMSG_LEN );

    return rc;
15  }

VOID WINAPI ModemGetLastError( PCHAR pBuf, DWORD nBuf )
{
20  strncpy( pBuf, ErrorMsg, nBuf );
```

CLAIMS

1. A communication system comprising:

a modem;

a communication channel;

5 the modem having internal settings representing communication parameters, the modem being communicatively coupled to the communication channel to carry out ongoing communications from the modem through the communication channel; and

10 a software module being associated with the modem, the software module accessing the internal settings of the modem via the communication channel and performing diagnostics using the internal settings of the modem.

2. The communication system of claim 1 wherein the software module further comprises a modem interface that interacts with the software module and assists the software module in performing diagnostics using the internal parameters of the modem.

15 3. The communication system of claims 1 or 2 wherein the software module accesses the communication channel transparently to the ongoing communications from the modem when the software module performs the diagnostics using the internal parameters of the modem.

20 4. The communication system of claims 1 wherein the software module accesses the communication channel without detrimentally affecting the ongoing communications across the communication channel.

5. The communication system of claims 1, 2, or 4 wherein the software module performs diagnostics using the internal parameters of the modem via the same communication channel that is used to carry out ongoing communications to and from the modem.

25 6. The communication system of claim 1 wherein the diagnostics performed by the software module comprise monitoring a data stream in the communication channel.

7. The communication system of claims 1, 2, 4, or 6 wherein the diagnostics performed by the software module comprise configuring the internal settings of the modem based on information obtained regarding a data stream to and from the modem.

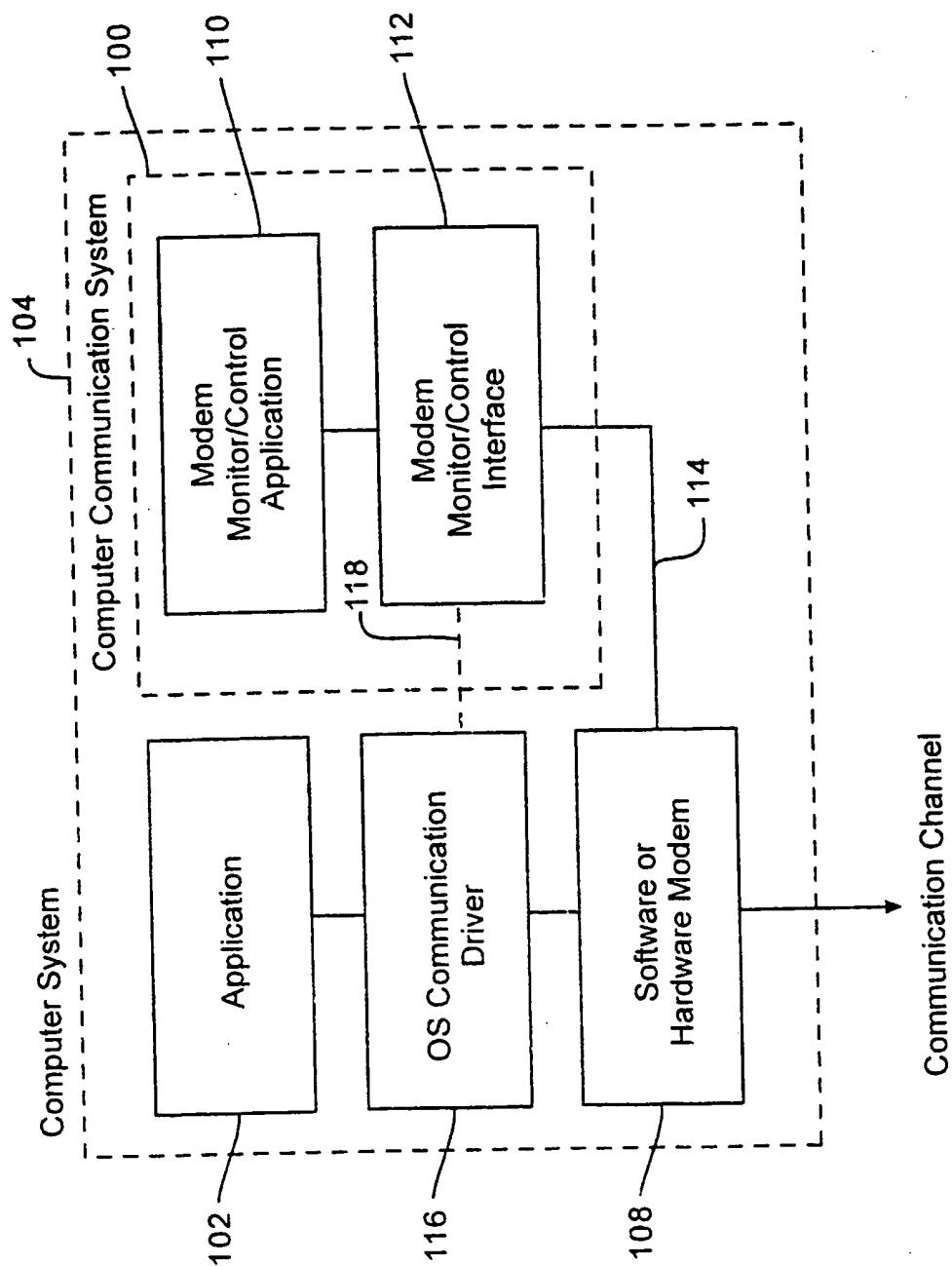
8. The communication system of claims 1, 2, 4, or 6 wherein the diagnostics 5 performed by the software module comprise controlling the internal settings of the modem according to information obtained regarding a data stream to and from the modem.

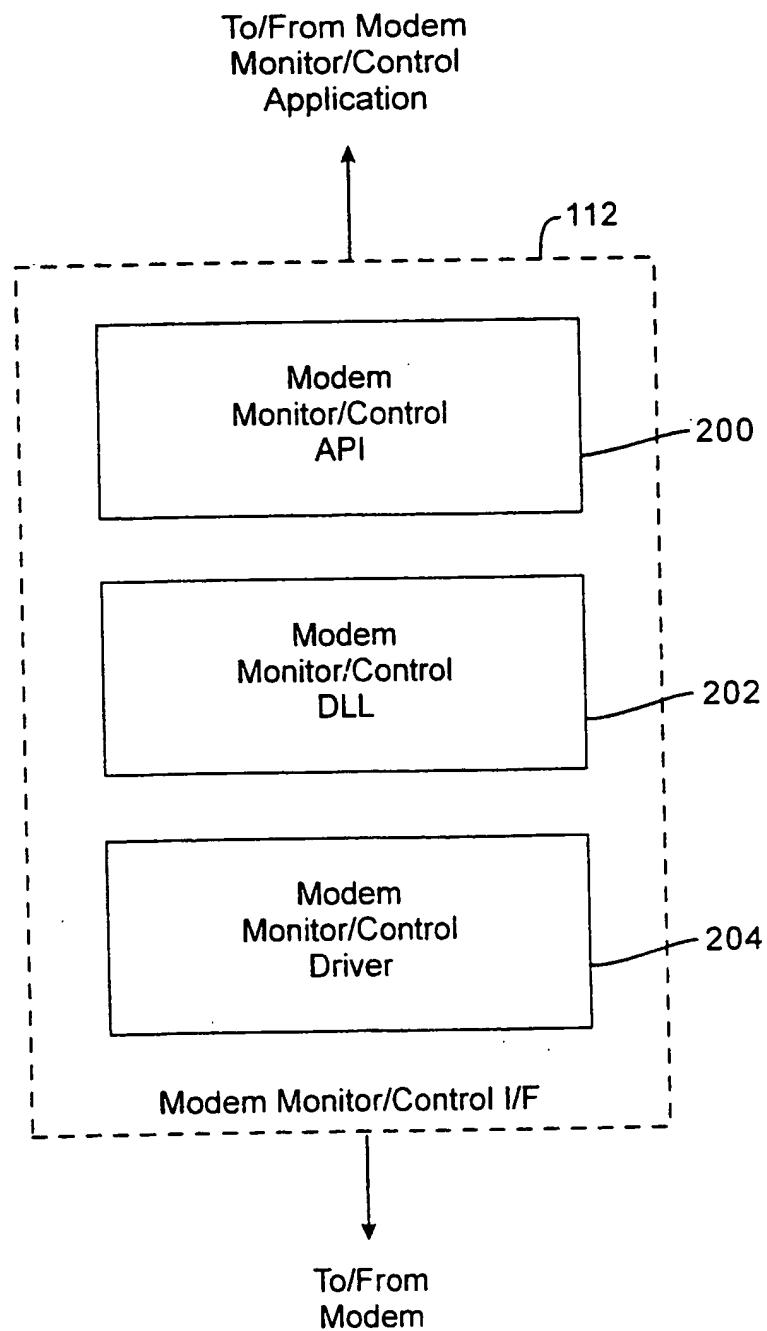
9. The communication system of claims 1, 2, 4, or 6 wherein the software module further comprises a user interactive interface for diagnostics.

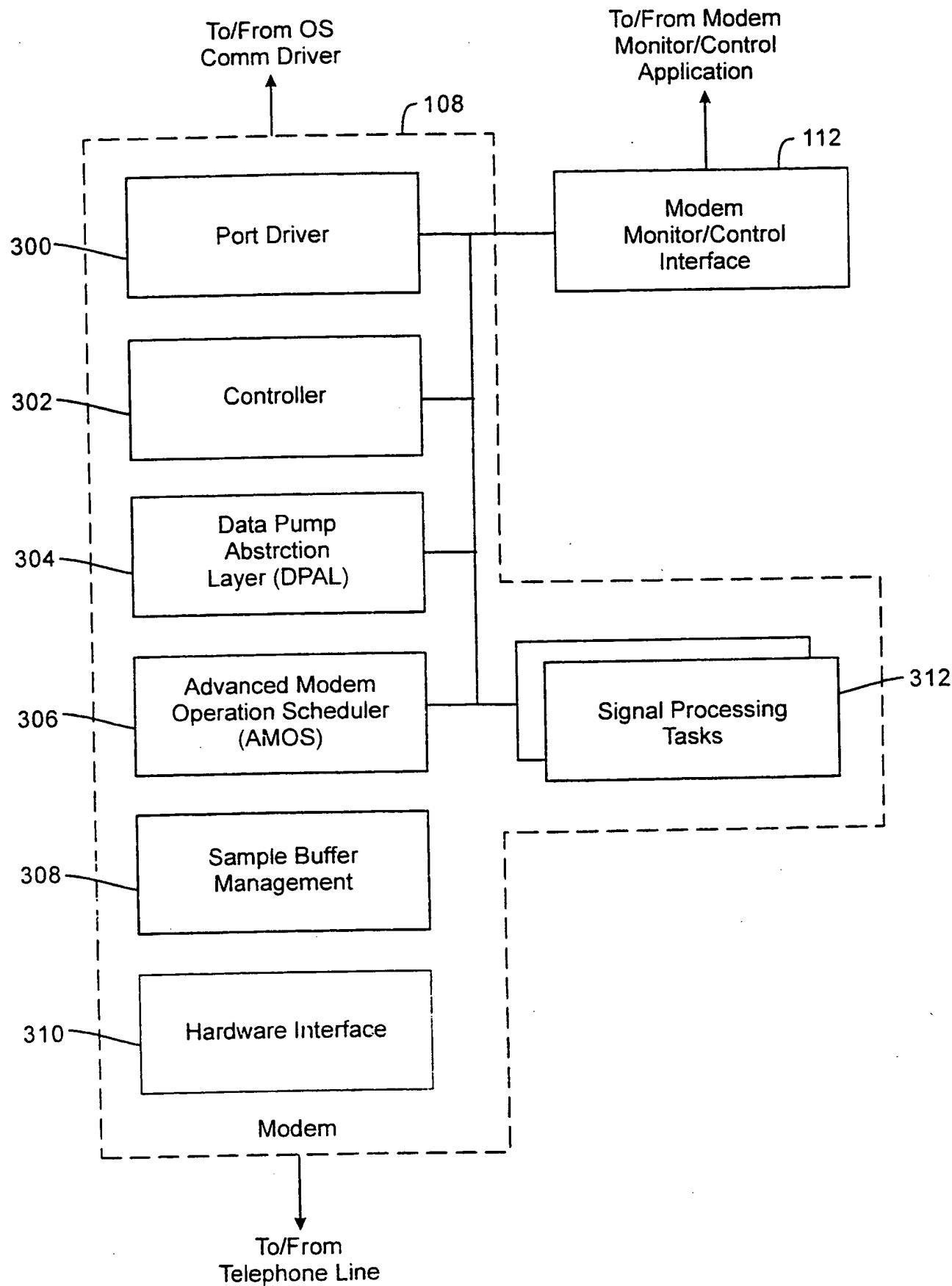
10. The communication system of claims 1, 2, 4, or 6 further comprising a plurality of software modules being associated, respectively, with each of a plurality of 10 modems.

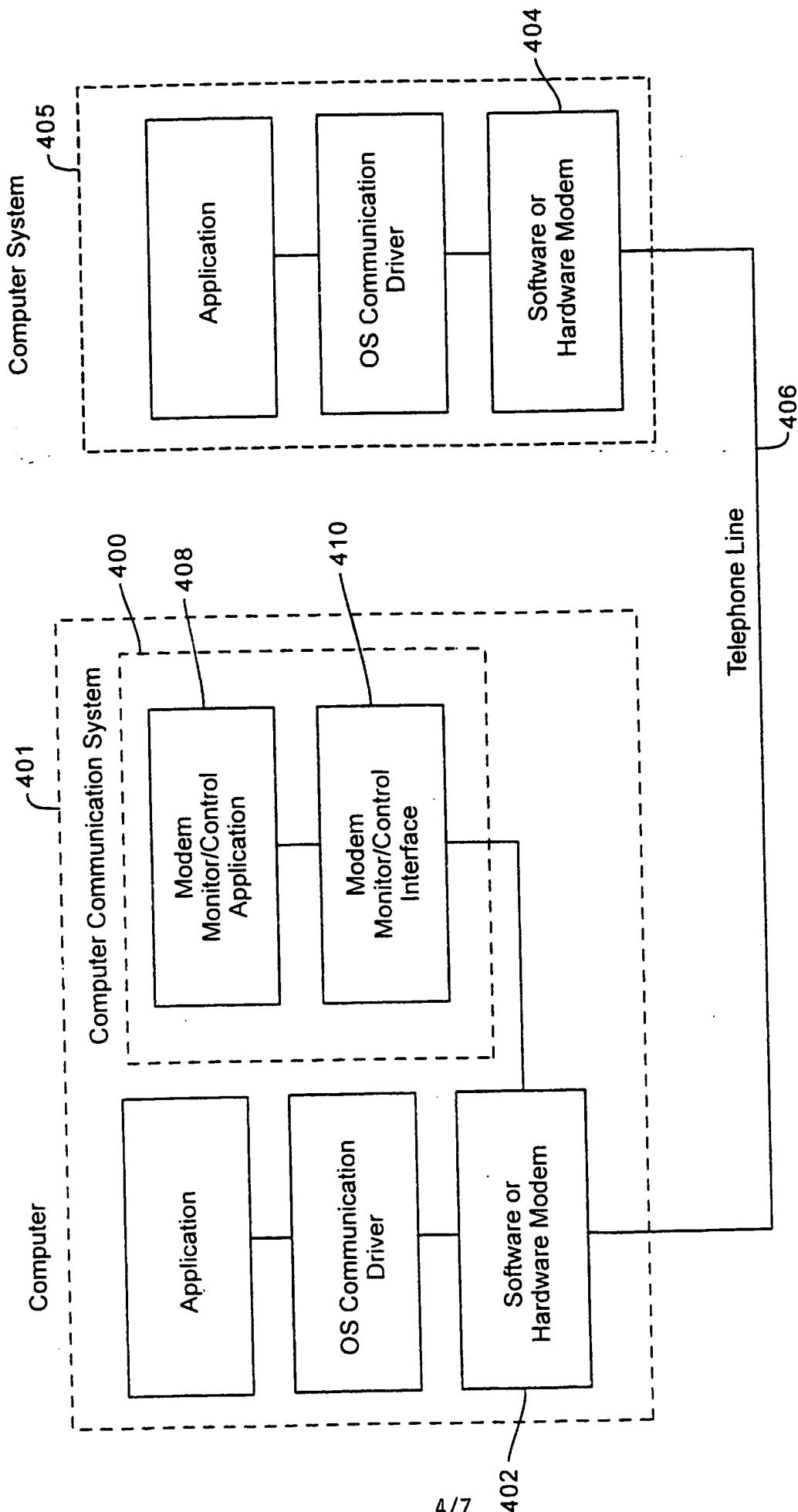
11. The communication system of claims 1, 2, 4, or 6 wherein the modem is communicatively coupled to the communication channel and thus to a network.

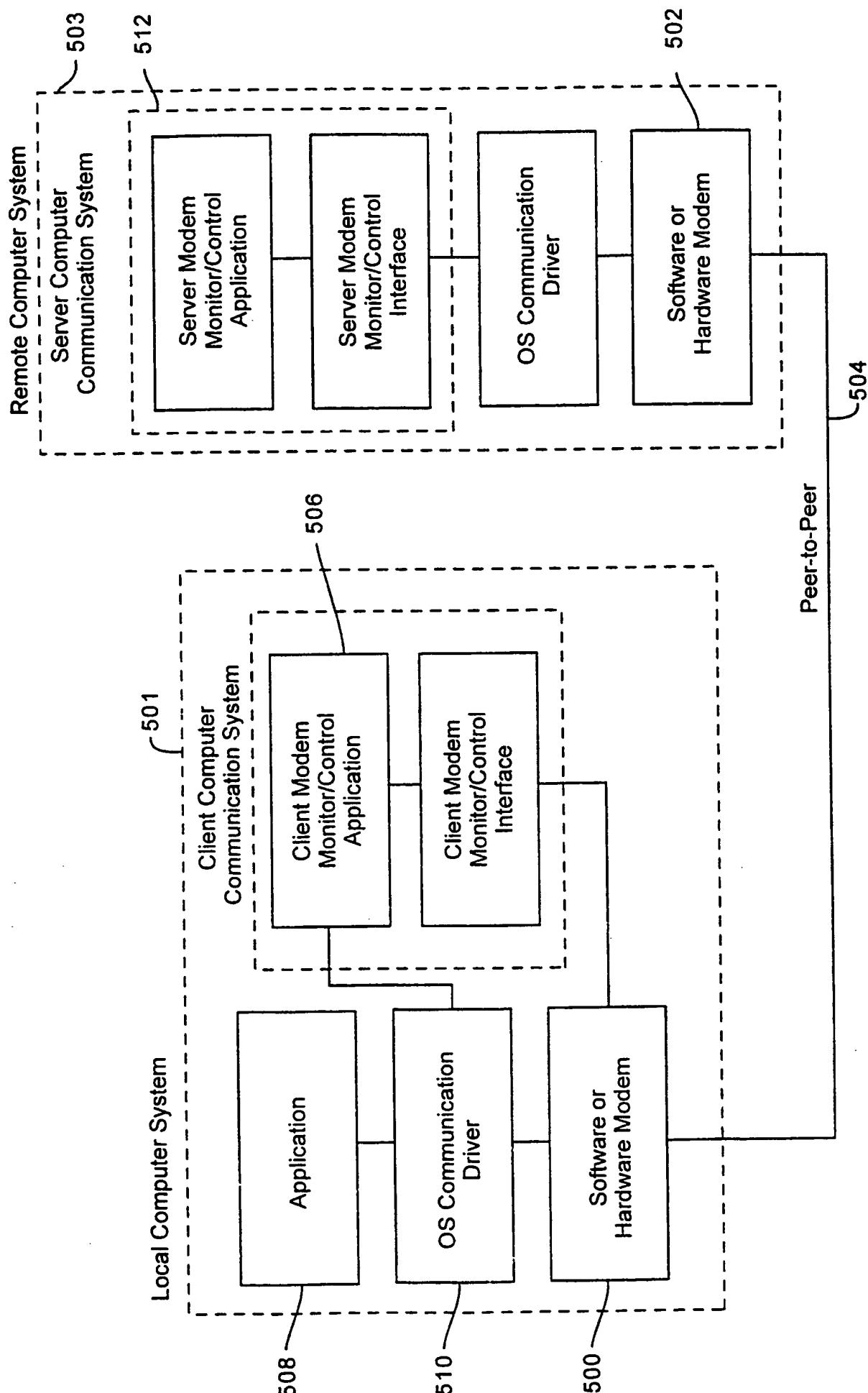
12. The communication system of claims 1, 2, 4, or 6 wherein the network is 15 selected from the group consisting of a local area network, a wide area network, and a global area network.

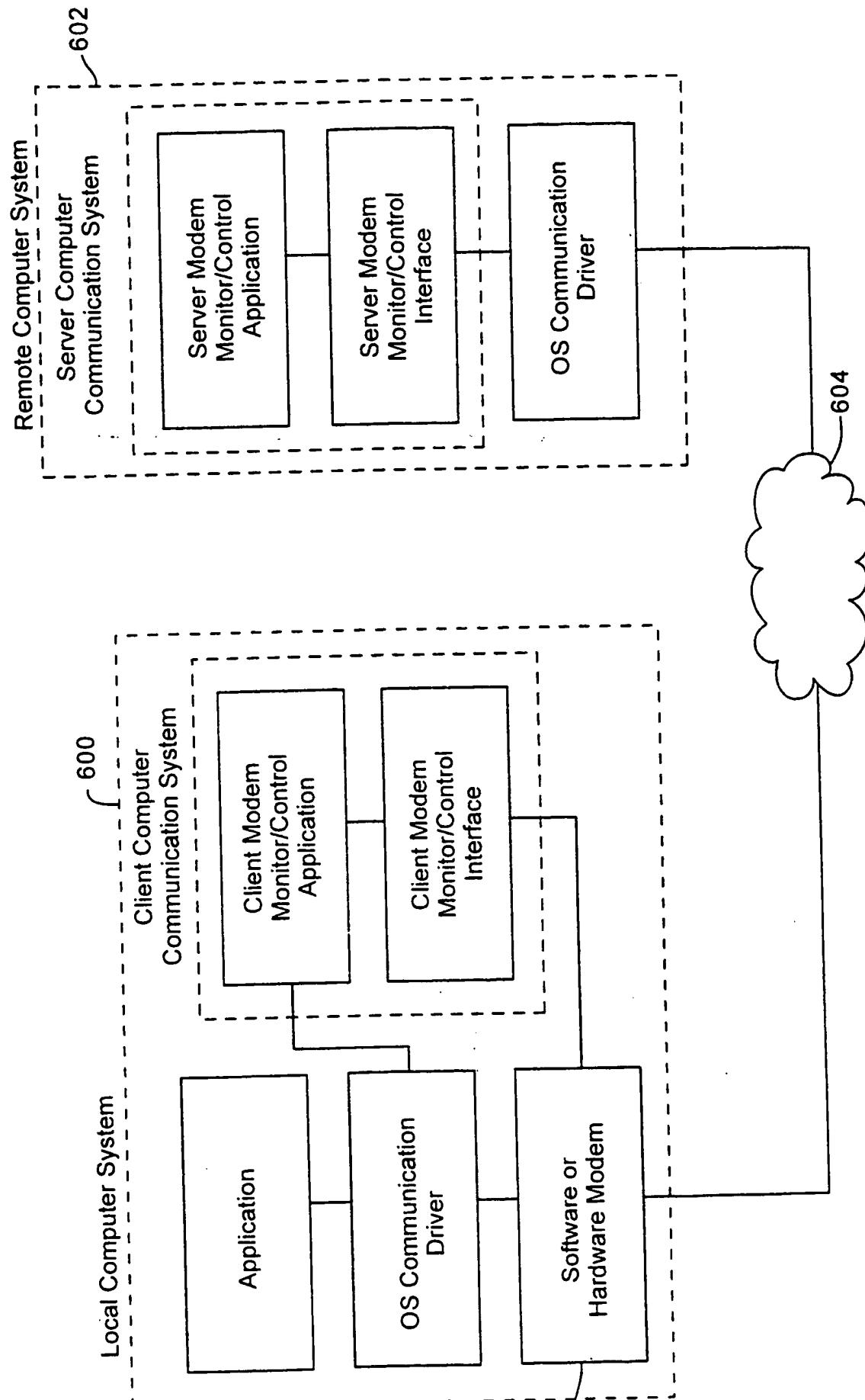


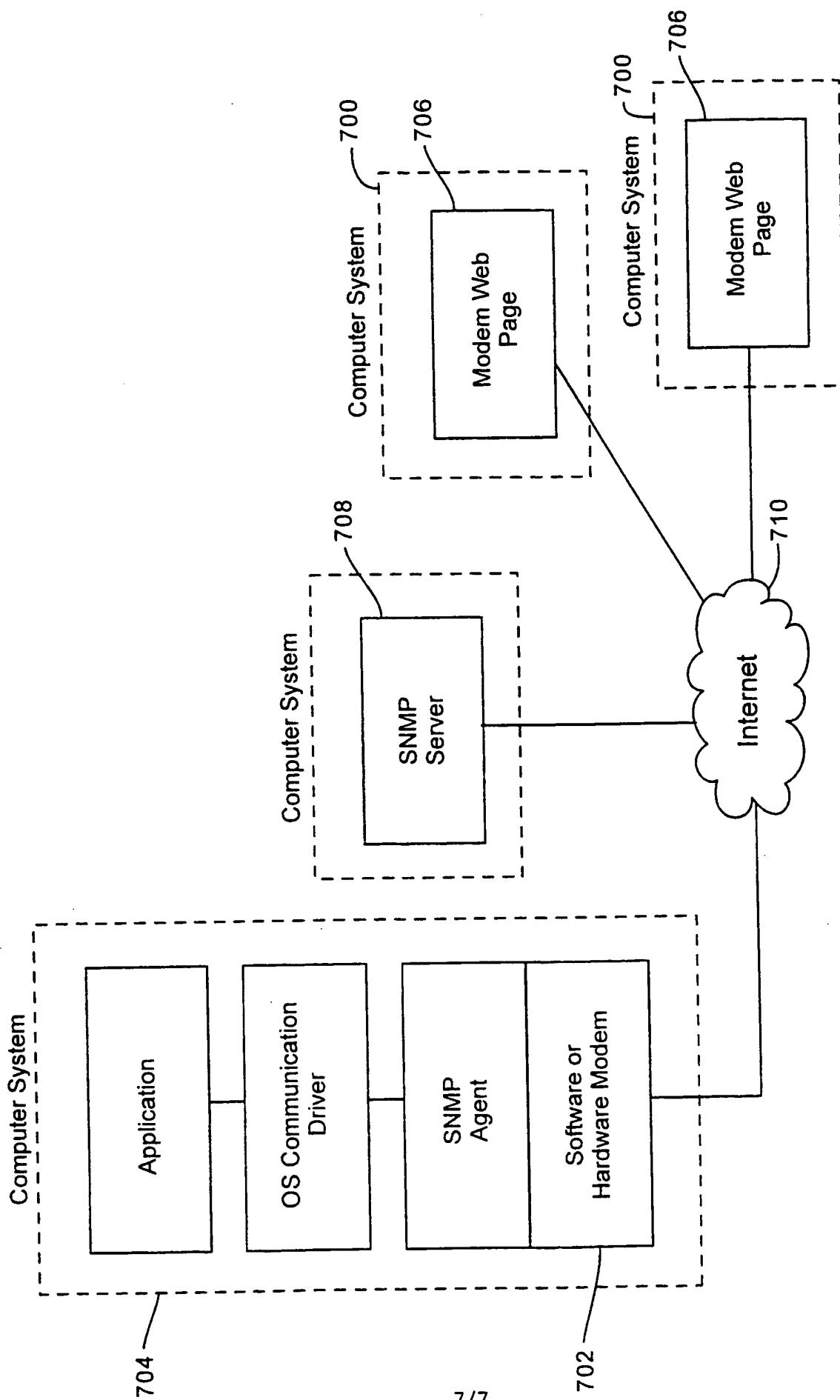












# INTERNATIONAL SEARCH REPORT

Invention: Application No  
PCT/US 99/04690

**A. CLASSIFICATION OF SUBJECT MATTER**  
IPC 6 H04M11/06 H04L12/26 H04L12/24

According to International Patent Classification (IPC) or to both national classification and IPC

**B. FIELDS SEARCHED**

Minimum documentation searched (classification system followed by classification symbols)

IPC 6 H04M H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

**C. DOCUMENTS CONSIDERED TO BE RELEVANT**

Category	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 4 516 216 A (ARMSTRONG THOMAS R) 7 May 1985 see abstract see column 1, line 10 - column 2, line 55 see column 3, line 18 - column 5, line 2 see column 7, line 40 - column 8, line 19 see figure 2 ---	1-4,6-12
Y	US 5 535 242 A (BRIGIDA DAVID J ET AL) 9 July 1996 see abstract see column 1, line 11 - column 3, line 21 see column 4, line 41 - column 6, line 9 see column 7, line 1-24 see figure 5 ---	5
		-/-

Further documents are listed in the continuation of box C.

Patent family members are listed in annex.

\* Special categories of cited documents :

- "A" document defining the general state of the art which is not considered to be of particular relevance
- "E" earlier document but published on or after the international filing date
- "L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
- "O" document referring to an oral disclosure, use, exhibition or other means
- "P" document published prior to the international filing date but later than the priority date claimed

- "T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
- "X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
- "Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
- "&" document member of the same patent family

Date of the actual completion of the international search

5 July 1999

Date of mailing of the international search report

27/07/1999

Name and mailing address of the ISA

European Patent Office, P.B. 5818 Patentlaan 2  
NL - 2280 HV Rijswijk  
Tel. (+31-70) 340-2040, Tx. 31 651 epo nl.  
Fax: (+31-70) 340-3016

Authorized officer

Lievens, K

## INTERNATIONAL SEARCH REPORT

Int'l. application No

PCT/US 99/04690

## C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 5 613 100 A (ANEZAKI AKIHIRO) 18 March 1997 see abstract see column 1, line 14 - column 3, line 4 see column 4, line 3 - column 8, line 49 see figures 1,3,21 ----	1-4, 10-12 5-9
X	"Dynamic Setting of Modem Parameters" IBM TECHNICAL DISCLOSURE BULLETIN, vol. 26, no. 1, June 1983, pages 261-262, XP002108167 US see the whole document ----	1,2, 10-12

## INTERNATIONAL SEARCH REPORT

Information on patent family members

Int'l. Appl. No. 99/04690  
PCT/US

Patent document cited in search report	Publication date	Patent family member(s)			Publication date
US 4516216	A 07-05-1985	NONE			
US 5535242	A 09-07-1996	US	5528626	A 18-06-1996	
		BR	9301307	A 05-10-1993	
		CA	2087507	A,C 01-10-1993	
		EP	0565229	A 13-10-1993	
		JP	2688157	B 08-12-1997	
		JP	6085878	A 25-03-1994	
		KR	9709296	B 10-06-1997	
US 5613100	A 18-03-1997	JP	2062619	C 24-06-1996	
		JP	3099351	A 24-04-1991	
		JP	7095313	B 11-10-1995	

THIS PAGE IS FOR THE USE OF THE  
BANKER ONLY